SANYO Easy Radio IC
LV24000 / LV24001 / LV24002

# APPLICATION NOTE

## AN2400S04@

PRELIMINARY

SANYO Electric Co., Ltd. Semiconductor System-Business Div. Microcomputer Business Unit.
1-1-1, Sakata Oizumi-Machi, Gunma, JAPAN

## REVISION HISTORY

***V0.0*** *09-March-2004*

- Initial version

***V0.1*** *26-March-2004*

- Changed IF S-curve phase
- Modified after reviewing with project group
- Add timing diagrams and counter 2 sequence
- Add VQLP40 packaging dimensions
- Adjust Supply voltage

***V0.2*** *04-May-2004*

- Added flowcharts
- TSSOP24 (225 mil) packaging dimension added
- Read diagram added
- Register definitions changed:
  - Register 100h - chip ID definitions changed:
    - 04: LV24002
    - 05: LV24000/LV24001
  - Register 102h - MSR_O bit is moved from bit 7 to bit 4
  - Register 106h - SWP_CNT bit becomes SWP_CNT_L (active low)
  - Register 108h - IRQ_LVL bit is inverted (1: active low - 0: active high)
  - Register 202h – DIR_AFC bit: description changed
  - Register 206h - IF_PM becomes IF_PM_L (active low)
  - Register 207h - AMUTE becomes AMUTE_L (active low)
  - Register 207h - CTRLB becomes TB_ON (Treble/Bass on)
  - Register 208h - FILTSW bit becomes AUTOSSR
  - Register 209h - Tone (bit [7:4]) and volume (bit [3:0]) levels are inverted
  - Register 20Ah - Beep frequencies (bit [7:6]) are reversed
  - Register 20Ah - BASS_L becomes BASS_PP
  - Register 20Bh - Soft audio mute moved from bit [7:5] to bit [4:2]. 8 control levels
  - Register 20Bh - Soft stereo moved from bit [4:2] to bit [7:5]. 8 control levels
  - Register 208h - ST_M bit is inverted (1: mono - 0: stereo)

- Applied DIR_AFC bit level to "Using the digital automatic frequency control"
- Divider factor of counter 2 changed from 16 to 2
- Volume, tone, treble, bass handling added
- Soldering chapter is removed
- Write/Read timing changed

***V0.3*** *27-Aug-2004*

*-Added Datasheet's Register*

- Register definitions changed:
  Register 102h:
  - Bit 6 defined (AFC_LVL)
  - Bit 5 defined (AFC_SPD)
  - MSR_O bit is moved from bit 4 to bit 7
  - Bit 4 becomes reserved (was MSR_O)
- Register 202h – bit 4, 1, 0 are reserved but need to be written with 1
- Fix error in pin out of LV24002-VQLP40 (missing LINE IN R/L, double LINE OUT R/L)
- Change the default frequency of stereo decoder clock from 37.5 kHz to 38 kHz.
- Add default values of the registers
- Change $V_{cc1}$ to $V_{STABI}$.
- $L_{1,}$ $L_{2,}$ $V_{STABI}$ pin become NC for VQLP40 packaging

- Rename IFCEN_SOC register to IF_CENTER
- Register 10Eh: Change level of read only bits 2, 4 (was 1, become 0)
- Register 10Dh: change field strength bit values when read
- Register 206h: add description of AGCSP-bit
- Change write/read 3-wire timing diagram for correct CLOCK-level
- All chip types pin out: Swapping LINE-OUT-R and LINE-OUT-R for all packages
- LV24001/LV24002 pin out: Swapping LINE-IN-R and LINE-IN-L pin for all packages

***V0.4*** *29-September-2004*

- Register 10Ch: changed level of read-only bit 7 (always 0)
- Register 10Eh: changed description of bits 1 and 0
- Register 206h: changed bit 5 from AFC_WS into STABI_BP because bit function changed
- Register 207h: changed bit 7 from PWR_LVL into AGC_SLVL because bit function changed
- Corrected description for bits 0, 1 and 3
- Changed default for RADIO_CTRL3 register from 18h into 98h and changed remarks
- Improved write/read 3-wire description with respect to driving the CLOCK low
- Adjusted some current figures
- Change default stereo decoder clock from 38 kHz to 38.3 kHz
- Added text to RESET AFC when setting a frequency
- Appendix B: Code improvement
- Appendix D: Changed SwOscLow to 30 and SwOscHigh to 200
- Appendix G: Code improvement for Scan algorithm and FindFmStation
- Changed flowcharts “Set Frequency” and “Scan Frequency”

## TABLE OF CONTENTS

# INTRODUCTION

The Sanyo LV2400x family of products introduces a new concept of FM radio IC's that offer great functionality combined with excellent performance and without the use of external components. To achieve these characteristics new principles of tuning and control of the FM radio have been developed. This application note is intended to give insight in the principles that give the LV2400x product its compelling specifications.

Typical tuners IC's use an external clock and a PLL to generate the tuned RF frequency. In the LV2400x series, software on the host takes care of tuning the IC. This results in a very fast tuning and maximum flexibility in controlling and configuring the IC: scanning the whole FM band in less than 2 seconds by weak reception (no station), setting a FM frequency within 150 ms[1].

For tuning to a radio station three oscillator frequencies are important: The RF frequency, the IF frequency and the stereo decoder frequency. All these frequencies are programmed by software by adjusting four predefined setting frequencies registers: a DAC control register for each frequency (fine-tuning) and a capacitor switch control (coarse-tuning) for the RF frequency due to its great frequency range.

The above mentioned registers need to be set to specific values in order to correctly receive a radio station. The mechanism used for this is a "set and measure" algorithm using smart interpolation to quickly find out the right register's value.

Measuring the frequency is done by counting pulses (p) generated by the oscillator (RF, IF or stereo decoder) during a known time (t). The actual frequency can than be easily determined by dividing the (p) by (t). If the required frequency is not correct, the frequency register is reprogrammed. The new value can be estimated accurately by knowing the behavior of the oscillator when changing settings. This way only a few steps are required to find the right setting. Once the right frequency is found it can be locked by internal AFC logic so drift due to temperature or voltage changes is avoided.

The following chapters of the application note go in more detail on how to control the LV2400x. For a quick overview of the software refer to APPENDIX G: FLOW CHARTS.

[1] Based on 8-bit microprocessor with instruction clock at 2 MHz, driving 3-wires bus with 3 GPIO pins.

# HOST HARDWARE REQUIREMENTS

Access to the LV2400x IC is done through the 3-wire bus:

CLOCK Data strobe, input to the LV2400x
NR_W Command (Write or read data), input to the LV2400x
DATA Bi-directional pin: input to the LV2400x when NR_W is high, output from the LV2400x when NR_W is low.

Therefore, the host CPU should supply 3 GPIO pins, configured as following:

CLOCK Output of the host
NR_W Output of the host
DATA Bi-directional:
Output of the host when NR_W is high (write to LV2400x mode).
Input of the host when NR_W is low (read from LV2400x mode).
*Optional requirement of DATA-line:*
*Possibility to generate interrupt.*

To reduce the software load during measuring of a frequency, a (8-bit/16-bit) timer is preferable. The timer generates interrupt when it rolls over and continues with counting until the control software disables it.
When the timer is not available, the control software has to perform a busy waiting loop of ±32 ms for frequency measurements.

When an external clock, which is supplied by the host hardware, is connected to CLK_IN pin of the LV2400x, the busy waiting loop can be reduced (depends on the CLK_IN frequency).

Also, to reduce the software load on the host CPU, the DATA pin can be configured as an interrupt pin after the LV2400x is appropriately set up. In this manner, the control software doesn't have to poll the LV2400x for radio events (like stereo/mono mode changing, signal strength drops, measuring with CLK_IN done…)

# BASIC INFORMATION

## Radio regions

The popular radio regions are summarized below:

| Region | Band limit | De-emphasis |
|---|---|---|
| Japan | 76 MHz - 90 MHz | 50 μs |
| Europe | 87.5 MHz – 108 MHz | 50 μs |
| USA | 87.5 MHz – 108 MHz | 75 μs |

## Display VS tuned frequency

Radio stations are marked by their displayed frequency. To receive a radio station, a radio receiver must be tuned so that it can lock the IF for demodulation.
For the LV2400x, the tuned frequency is the sum of the displayed frequency and the preset IF frequency, in formula:

Tuned FM frequency = displayed frequency + preset IF frequency

For example: when the IF frequency of LV2400x is preset at 110 kHz, it must be tuned at 88.51 MHz to receive the radio station at 88.4 MHz.

# BASIC ROUTINES

## Accessing the LV2400x

Access to the LV24000x can be done through the 3-wire bus. At host side, The NR_W and CLOCK are output signals, while the DATA is bi-directional.

When power up, host should initialize the 3-wire bus in host read mode:

1. Set direction of the DATA-line to input-mode.
2. Drive NR_W low
3. Drive CLOCK high

**Note** : Use following sequence for changing read/write mode:

- A. Change from host read-mode to host write-mode:
  - A.1. Keep the CLOCK signal HIGH.
  - A.2. Set the NR_W signal to HIGH (write mode)
  - A.3. Set the DATA-pin direction to OUTPUT mode.
- B. Change from host write-mode to host read-mode:
  - B.1. Keep the CLOCK signal HIGH.
  - B.2. Set the DATA-pin direction to INPUT mode.
  - B.3. Set NR-W to LOW (read mode).

### *Writing the LV2400x*

Writing the LV2400x is done in two phases (if appropriate). First the correct block address needs to be written to the Block Select register (BLK_SEL). The 16-bits data pattern for the block selection consists of:

- Bit[15:8] = 0x01: the address of BLK_SEL register: block select cycle
- Bit[7:0] = block number: block to be selected

Note that the block select register needs to be written unless the last read of write was already done from/to the same block.

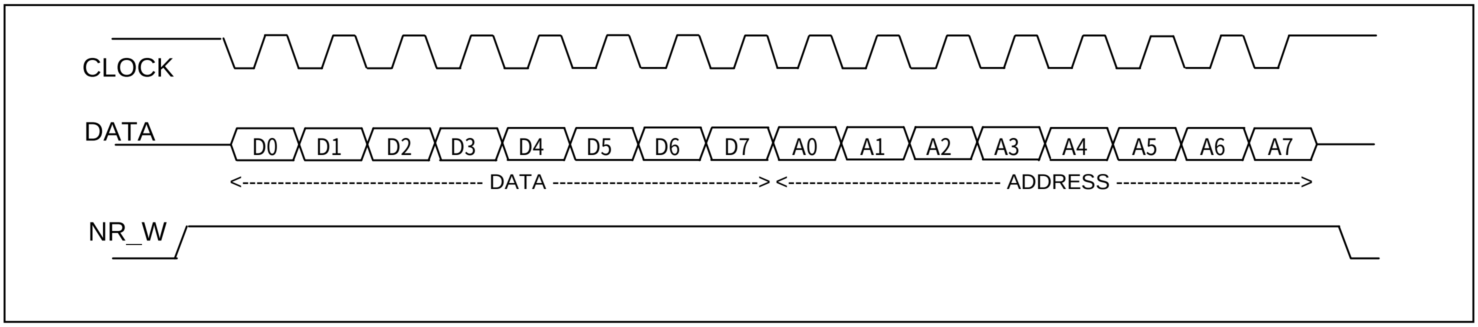
Next the register can be written. The 16 bits data pattern consists of:

- Bit[15:8]: 8 bits register address.
- Bit[7:0]: 8 bits register data.

The 16-bits data pattern (block select and register write) is serially sent to the LV2400X as follows:

a) Drive NR_W pin high to set the LV2400X in input mode.
b) Set the direction of DATA-line to output of the host (if required)
c) Generate negative edge of CLOCK (driving CLOCK from high to low)
d) Drive the DATA pin to correct level.
e) Generate positive edge of CLOCK (driving CLOCK from low to high). This signals the LV2400x to latch the data bit.
f) Delay some time to meet the data hold time requirement of LV2400X.
g) Repeat step (c) to (f) 16 times to shift the 16 bits data pattern into the LV2400X.
h) Set the direction of DATA-line to input of the host (if required)
*i)* Drive NR_W pin to low. This signals the LV2400X to latch the data into correct register.

**Note**: LSB of the data pattern should be shifted out first.


CLOCK
DATA
D0
D1
D2
D3
D4
D5
D6
D7
A0
A1
A2
A3
A4
A5
A6
A7
<--------------------------------- DATA ----------------------------> <----------------------------- ADDRESS ------------------------->
NR_W

### *Reading the LV2400X*

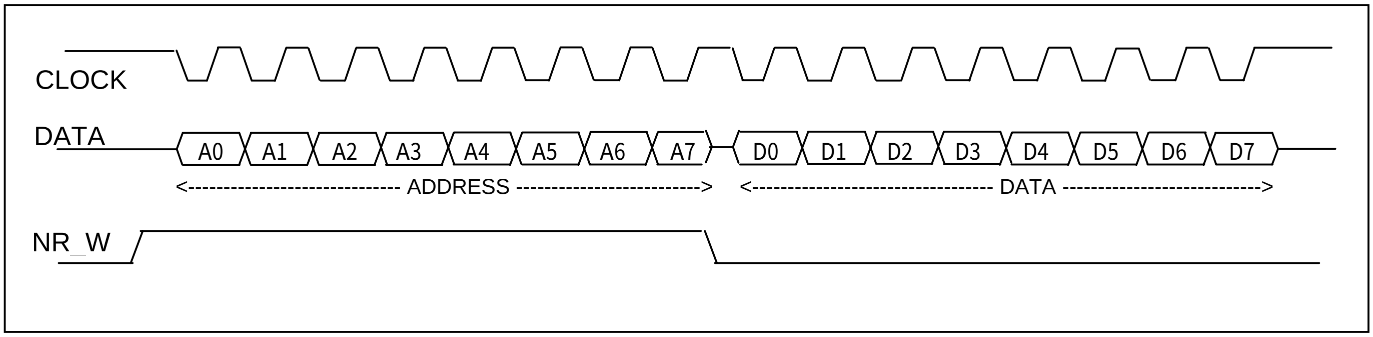
To read a chip register, the 8 bits register address must be written followed by reading the 8 data bits. Remember that before accessing a register, the correct block must be selected first, unless it already selected (see above).

The register can be serially read as follow:

a) Optionally select correct block
b) Drive NR_W line high (write mode)
c) Set the direction of DATA-line to output of the host (if required)
d) Generate negative edge of CLOCK (driving CLOCK from high to low)
e) Drive DATA-line to correct level
f) Generate positive edge of CLOCK (driving CLOCK from low to high).
g) Delay some time to meet the data hold time requirement of LV2400X.
h) Repeat step (d) to (i) 8 times to shift the 8 bits register address into the LV2400X (LSB must be shifted out first)
i) Set the direction of DATA-line to input of the host (if required)
j) Drive the NR_W line low (read mode).
k) Drive the CLOCK high to low to generate a negative edge. This signals the LV2400x to put the data bit on the DATA-line.
l) Delay sometime to meet the data setup time requirement of LV2400x.
m) Drive CLOCK-line low to high.
n) Read the data bit at the DATA-line (The data can also be read after step l)
o) Repeat step (k) to (n) 8 times to read the 8 bits data out of the LV2400x.

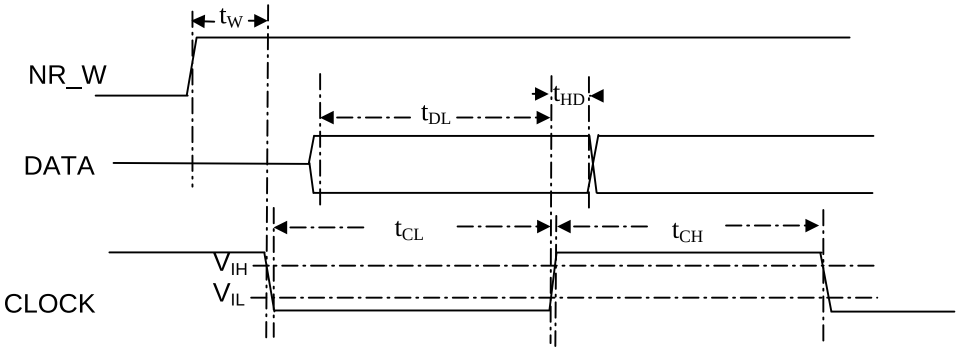
**Note**: The LV2400X will shift the LSB of the data out first.

**Note**: The LV2400X will shift the LSB of the data out first.


CLOCK
DATA
A0
A1
A2
A3
A4
A5
A6
A7
D0
D1
D2
D3
D4
D5
D6
D7
<----------------------------- ADDRESS ------------------------->
<--------------------------------- DATA --------------------------->
NR_W

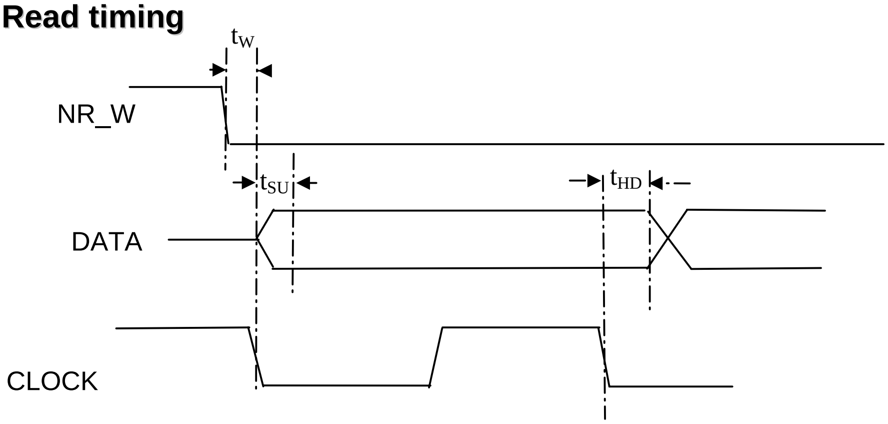
## Timing diagrams

### Write timing



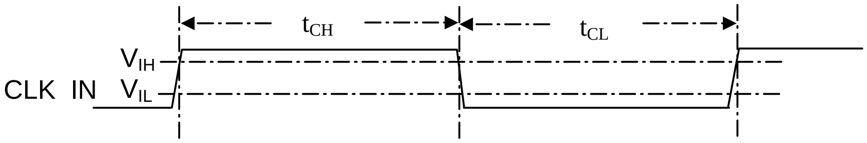
| Parm | | Ratings | | | Unit |
|---|---|---|---|---|---|
| | | Min. | Typ. | Max. | |
| $t_W$ | Delay from command to data | 750 | | | ns |
| $t_{DL}$ | Delay from data stable to data latch time | 750 | | | ns |
| $t_{HD}$ | Data Hold time | 750 | | | ns |
| $t_{CH}$ | Clock High-level time | 750 | | | ns |
| $t_{CL}$ | Clock Low-level time | 750 | | | ns |

### Read timing



| Parm | | Ratings | | | Unit |
|---|---|---|---|---|---|
| | | Min. | Typ. | Max. | |
| $t_W$ | Delay from command to 1st data bit | 350 | | | ns |
| $t_{SU}$ | Data Setup time | | | 350 | ns |
| $T_{HD}$ | Data hold time | | | 350 | ns |

## External clock timing

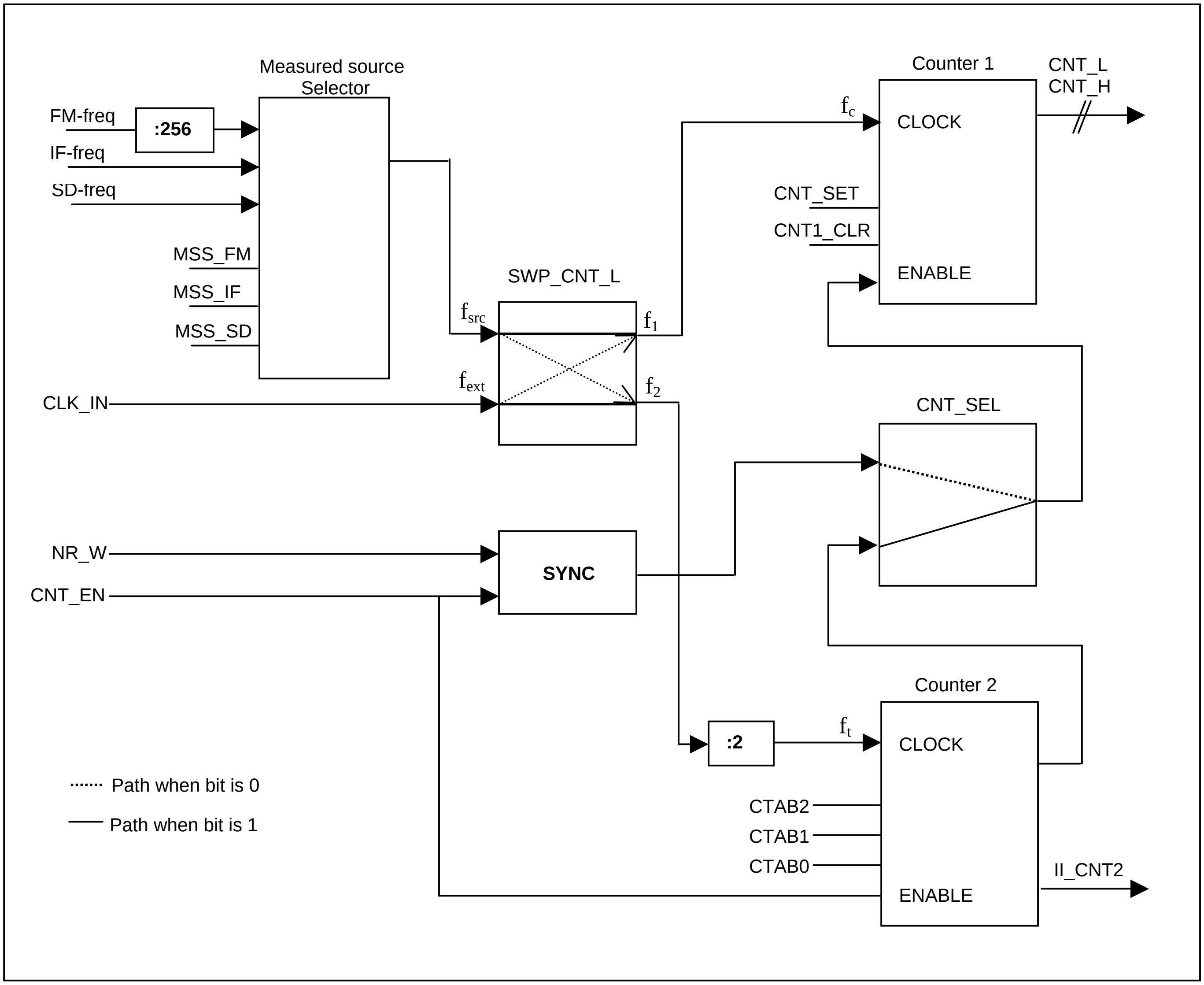


| Parm | | Ratings | | | Unit |
|---|---|---|---|---|---|
| | | Min. | Typ. | Max. | |
| $t_{CH}$ | Clock High-level time | 495 | | | ns |
| $t_{CL}$ | Clock Low-level time | 495 | | | ns |

# Measure frequency

## Measuring frequency with the LV2400X

The 3 frequencies IF, stereo decoder clock and FM can be determined by counting the pulses within a timing window. The pulses can be counted with the built-in counter 1. The timing window can be created by host software or by (optionally) using counter 2.
When counter 1 is selected (CNT_SEL bit in CNT_CTRL register is 0), the measurement is controlled by NR_W-line: counter 1 starts counting when it is enabled and the NR_W-line goes low. Counter 1 stops with counting as soon as the NR_W-line goes high. The 16-bit pulse count can be read back at CNT_H/CNT_L register. The active time of NR_W is the measuring period.



When counter 2 is selected, the measurement is controlled by CLK_IN line and the tab select bits CTAB[2:0]. Counter 2 will enable counter 1 when CNT_EN is active, after the number of count which is selected by the host software via CTAB[2:0]-bits, counter 2 stops the measurement and drives II_CNT2 flag active to indicate the measurement is ended. The 16-bit pulse count can be read back at CNT_H/CNT_L register. The input clock of counter 2 and the tab-selection determines the measuring period. When SWP_CNT_L bit is high, the measuring source will go to counter 2 instead of counter 1. Only clear this bit when CLK_IN is greater than 100 kHz.

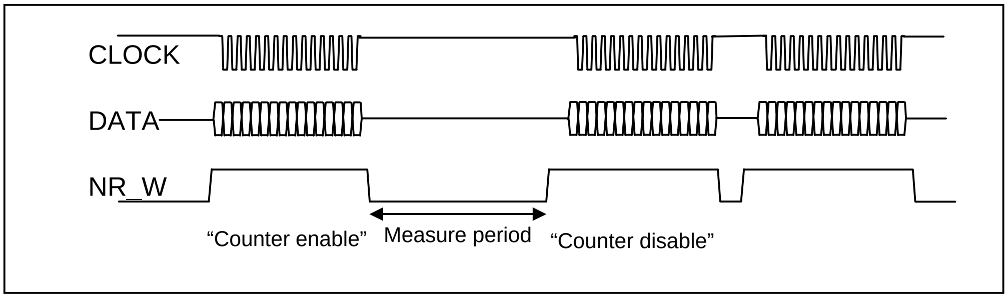
The frequencies of the LV2400x are divided as below table before they go to the measuring circuitry:

| Frequency | Divider factor |
|---|---|
| IF-frequency | 1 |
| RF-frequency | 256 |
| Stereo decoder clock | 1 |

***Measuring with counter 1 (NR_W control)***

Perform following steps:

a) Enable the frequency source to be measured (register MSRC_SELL – set one of the MSS_xx bits).
b) Make sure counter 1 is selected (Register CNT_CTRL – bit CNT_SEL is 0)
c) Enable measuring mode (register RADIO_CTRL1 – EN_MEAS bit).
d) Reset the counter (register CNT_CTRL – Driving bit CNT1_CLR high then low).
e) Start the counter 1 on LV2400x (register CNT_CTRL – set CNT_EN bit). At the moment the NR_W signal gets LOW, the counter starts.
f) Wait time t.
g) To stop the counter, first set the NR_W signal HIGH, then disable the counter of LV2400x (register CNT_CTRL – clear CNT_EN bit).
h) Read the pulse count n from the counter register of LV2400x (register CNT_H/CNT_L).
i) Restore the measure mode.
j) Restore the measure source select (register MSRC_SEL)



**Note:**

- *The measuring window begins at the moment that the NR_W signal is driving LOW (point e) and ends when the NR_W signal is driving HIGH (point g).*
- *The precision of the measurement depends on:*
    a) *The duration of t. 1 pulse wrong at t=1ms results in more deviation than at t=32ms*
    b) *The precision of the measuring window: calculate with t=32ms gives other f than with t=32,1ms. Application should take some care to have an accurate measuring window t.*

Then the frequency can be calculated with formula:

$$\text{Frequency [Hz]} = \frac{\text{Count value}}{\text{Counting time [s]}} \; x \; \text{Divider factor}$$

The accuracy of this method is shown in below table.

| Measure period | Measure deviation | | |
|---|---|---|---|
| | IF-frequency | RF-frequency | Stereo-decoder frequency |
| 8 ms | ± 125 Hz | ± 32 kHz | ± 125 Hz |
| 16 ms | ± 62 Hz | ± 16 kHz | ± 62 Hz |
| 32 ms | ± 31 Hz | ± 8 kHz | ± 31 Hz |
| 64 ms | ± 15 Hz | ± 4 kHz | ± 15 Hz |
| 100 ms | ± 10 Hz | ± 2.5 kHz | ± 10 Hz |

***Measuring with counter 2 (CLK_IN control)***

Perform following steps:

a) Enable the frequency source to be measured (register MSRC_SELL – set one of the MSS_xx bits).
b) Make sure counter 2 is selected (Register CNT_CTRL – bit CNT_SEL is 1)
c) Enable measuring mode (register RADIO_CTRL1 – EN_MEAS bit).
d) Reset the counter (register CNT_CTRL – Driving bit CNT1_CLR high then low).
e) Program the appropriate tab-select (register CNT_CTRL - CTAB[2:0] bits)
f) Program the SWP_CNT_L bit in CNT_CTRL register appropriately
g) Enable counter 2 interrupt (register IRQ_MSK – IM_CNT2 bit)
h) Start the counter on LV2400x (register CNT_CTRL – set CNT_EN bit).
i) Write any data pattern to IRQ_OUT register to let the LV2400x use the DATA-line as interrupt
j) Host software can poll the DATA-line or wait for interrupt.
k) When counter 2 interrupt occurs (DATA-line goes active, II_CNT2 flag is set in IRQ_ID register), de measuring is done.
l) Disable the counter of LV2400x (register CNT_CTRL – clear CNT_EN bit).
m) Read the pulse count n from the counter register of LV2400x (register CNT_H/CNT_L).
n) Restore the measure mode.
o) Restore the measure source select (register MSRC_SEL)
p) Restore interrupt setting

The frequency can be calculated as follows:

When SWP_CNT_L is 1 (no counters swapping):

$$\text{Frequency [Hz]} = \frac{N * f_{ext}}{Tab * 2} \; x \; \text{Divider factor}$$

When SWP_CNT_L is 0 (counters are swapped):

$$\text{Frequency [Hz]} = \frac{f_{ext} * Tab * 2}{N} \; x \; \text{Divider factor}$$

*N: pulse count (read back from CNT_L/CNT_H)*
*$f_{ext}$: Frequency of the external clock on CLK_IN-line*
*Tab: tab selected by CTAB[2:0] (example: if CTAB[2:0] = 101b, value of tab is 2048)*

The deviation 1/N (assume no deviation in $f_{ext}$)

## Using the Digital Automatic Frequency Control (AFC) of the LV2400x

AFC is the mechanism that prevents the FM-frequency from drifting (FM-frequency drifting will de-tune the radio reception)

To enable the AFC:

- The AFC_WS bit (RADIO_CTRL2 register) should be high
- Clear the DIR_AFC bit (RADIO_CTRL1 register) for normal operation mode
- Clear the RST_AFC bit ((RADIO_CTRL1 register)
- Set the EN_AFC bit (RADIO_CTRL1 register)

To disable the AFC:

- Don’t touch the AFC_WS bit (RADIO_CTRL2 register) and DIR_AFC bit (RADIO_CTRL1 register)
- Set the RST_AFC bit (RADIO_CTRL1 register)
- Clear the EN_AFC bit (RADIO_CTRL1 register)

Because the AFC adjusts the FM-frequency, it is recommended to disable the AFC before setting FM-frequency.

## Using interrupt of the LV2400x

Prepare the LV2400x for generating interrupt:
a) Clear any pending interrupt (by reading RADIO_STAT and CTRL_STAT register)
b) Program the interrupt level (register IRQ_MSK - IRQ_LVL bit)
c) Program the IRQ_MSK register to enable the desired interrupt(s)
d) Write any data pattern to IRQ_OUT register to let the LV2400x generate interrupt on the DATA-line

Interrupt handler on the host side:
a) Read the IRQ_ID register to identify the interrupt(s)
b) Serve all enabled interrupts
c) Clear the served interrupt(s)
e) Write any data pattern to IRQ_OUT register to arm the interrupt again

Overview of LV2400x interrupts

| Interrupt | Enable bit (IRQ_MSK) | ID bit (IRQ_ID) | Clear action | Handling |
|---|---|---|---|---|
| Counter 2 done | IM_CNT2 | II_CNT2 | Disable counter | Frequency calculation |
| AFC out of range | IM_AFC | II_AFC | Read CTRL_STAT register | Re-tune the FM-frequency |
| Mono/Stereo changed | IM_MS | II_FS_MS | Read RADIO_STAT register | Update display |
| Field strength changed | IM_FS | II_FS_MS | Read RADIO_STAT register | Update display |

## Using audio control of the LV2400x

### *Audio volume control*

21 volume levels can be realized using AMUTE_L-bit, VOLSH-bit in RADIO_CTRL3-register and the 4 volume level VOL_LVL bits in AUDIO_CTRL1-register as following scheme:

| Volume | AMUTE_L | VOLSH | VOL_LVL[3:0] | Remark |
|---|---|---|---|---|
| 0 | 0 | X | X | No sound (audio muted) |
| 1…16 | 1 | 0 | 15…0 | Volume without VOLSH-bit |
| 17…20 | 1 | 1 | 3..0 | Extra levels with VOLSH-bit |

### *Tone (loudness) control*

The 4 tone level bits TONE_LVL (bit [7:4] of the AUDIO_CTRL1-register) can be used for dynamic bass boost feature: Host software can let the tone level follow the volume level until a pre-defined tone level is reached to introduce more or less bass according to the volume level. Program these 4 bits with 1111b to keep the LV2400x at fixed bass level when this feature is not desired (no dynamic bass boost).
Host software should make sure that the tone level may not exceed the volume level (Note that tone/volume levels are the inverse of the register's value). For example when the dynamic bass boost is preset at 9 (value 15-9=6 must be used as lowest tone value), following scheme should be used:

| VOL_LVL (AUDIO_CTRL1[3:0]) | TONE_LVL (AUDI_CTRL1[7:4]) | Remark |
|---|---|---|
| 15...6 | 15...6 | Tone bits follow volume bits (Volume level varies from 0…9, so does tone level) |
| 5…0 | 6 | Tone bits stick at 6 for dynamic bass level 9 |

### *Treble control*

3 treble levels can be realized using TREB_N-bit, TREB_P-bit in AUDIO_CTRL2 -register and TB_ON-bit in RADIO_CTRL3-register as following scheme:

| **TB_ON** | **TREB_N** | **TREB_P** | **Remark** |
|---|---|---|---|
| 1 | 1 | 0 | Treble level 0 |
| 0 | 0 | 0 | Treble level 1 (flat frequency response) |
| 1 | 0 | 0 | Treble level 1 (do not use) |
| 1 | 0 | 1 | Treble level 2 |

**Note**: Not mentioned combinations are not allowed.

### *Bass control*

4 bass levels can be realized using BASS_N-bit, BASS_P-bit, BASS_PP-bit in AUDIO_CTRL2 -register and TB_ON-bit in RADIO_CTRL3-register as following scheme:

| **TB_ON** | BASS_N | BASS_P | BASS_PP | |
|---|---|---|---|---|
| 1 | 1 | 0 | 0 | Bass level 0 |
| 0 | 0 | 0 | 0 | Bass level 1 (flat freq. response) |
| 1 | 0 | 0 | 0 | Bass level 1 (do not use) |
| 1 | 0 | 1 | 0 | Bass level 2 |
| 1 | 0 | 1 | 1 | Bass level 3 |

**Note**: Not mentioned combinations are not allowed.

# BASIC THEORY

## Redefine the LV2400x registers for software ease

### *Dealing with negative DAC control registers*

The LV2400x has some DAC control registers in negative logic (i.e. TBD, TBD). This means when increasing the content of those registers, the frequency is decreased.
The FM_OSC (TBD) register is on the contrary, positive logic. Software can use the following conversion to convert the negative DAC control registers to positive logic:
*Logical value (SoftwareValue) to physical value (HardwareValue) conversion:*

*SoftwareValue = 255 – HardwareValue*

*Physical value (HardwareValue) to logical value (SoftwareValue) conversion:*

*HardwareValue = 255 – SoftwareValue*

Applying the conversion on the negative DAC control registers, software can work with the logical value to have all the DAC control registers of LV2400x in positive logic. A low-level routine converts the logical value to physical value before writing it to the hardware.
Having all DAC control registers in positive logic makes it possible for the software to use just one algorithm for determining the DAC control register value by a frequency.

### *Dealing with the 7½ bits FM_CAP register*

Working with the FM_CAP register requires some care because:

- It's in negative logic.
- The value range is not continuous due to the overlapping of bit 7 and bit 6 of this register (combination 01b and 10b have the same range)

Software can apply following conversion to the FM_CAP:
*Logical value (SoftwareValue) to physical value (HardwareValue) conversion:*

```
If (SoftwareValue <64)
        HardwareValue = 255 – SoftwareValue
Else
        HardwareValue = 255 – 64 – SoftwareValue
```

*Physical value (HardwareValue) to logical value (SoftwareValue) conversion:*

```
If (HardwareValue <128)
        SoftwareValue = 255 – 64 – HardwareValue
Else
        SoftwareValue = 255 – HardwareValue
```

After the conversion, software will get a linear FM_CAP register in positive logic. The logical value range is from 0 to 191.

## Finding out the value for a DAC control register

Tuning the LV2400x to a frequency is finding out which value should be programmed in the companion DAC control register.
There are 3 tunable frequencies on the LV2400x:

- IF frequency: controlled by the IF_OSC register.
- Stereo decoder clock: controlled by the SD_OSC register.
- RF frequency: controlled by the FM_CAP (coarse step) and FM_OSC (fine step) register.

These 4 registers can be tuned with 1 algorithm (see APPENDIX B: BIG STEP TUNING). The fine step tuning (see APPENDIX C: FINE STEP TUNING) can be optionally used when precision is needed.

During radio station scanning, the FM_CAP and FM_OSC must be adjusted to vary the RF frequency. So the values for these 2 registers should be quickly determined for a quick scan. Software can approach them via their math model, as described in next session.

## Quick setting RF frequency

Setting the RF frequency is finding out the values for FM_CAP and FM_OSC register. The FM_CAP adjusts the RF frequency in big steps. The FM_OSC adjusts the RF frequency in small steps.

The RF frequency is generated with formula:

$\omega^2 = 1/(LC)$ where $\omega = 2\pi f$ (f is the desired RF)
L is the inductance of the coil
C is the capacitance (controlled by FM_CAP/FM_OSC)

Or

$C = 1/(L\omega^2)$
$C = 1/(L*4\pi^2) * 1/f^2$

The first part of the equation ($1/(L4\pi^2)$), is a coefficient which varies with L but L can be considered as constant for a specific LV2400x in a specific hardware design. So when we redefine this part as a constant A (see further APPENDIX H: MISCELLANEOUS ROUTINES - Constant A for RF-frequency coefficient), a coefficient for a frequency f can be calculated as follows:

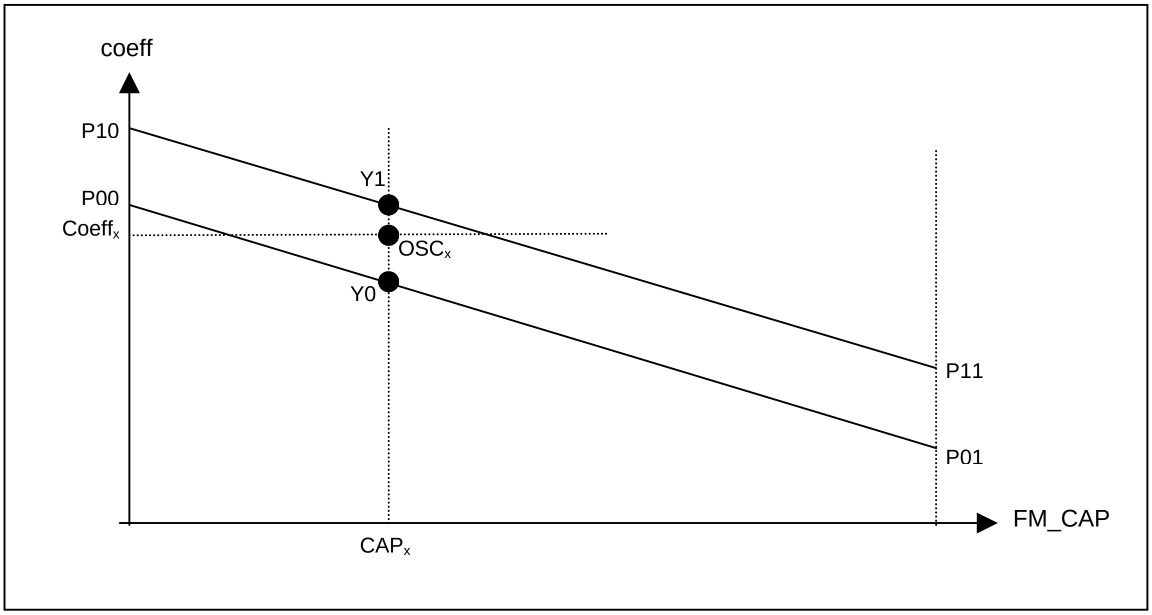
$Coeff = (A/f^2)$

When we use the coefficient to approach the RF frequency, we will have a linear characteristic so that a FM_CAP value for a RF frequency can be interpolated between the point FM_CAP=low and the point FM_CAP=high.

When we apply the FM_OSC to the FM_CAP, we will have 4 points, divide into 2 sets as follows:

| FM_OSC | FM_CAP | Point |
|---|---|---|
| Low | Low | P00 (set 0) |
| Low | High | P01 (set 0) |
| High | Low | P10 (set 1) |
| High | High | P11 (set 1) |

In graphic:



With these 4 points, the FM_CAP and FM_OSC of a RF-frequency $f_x$ can be calculated as follows:

- Calculate $coef_x$ from RF frequency $f_x$ ( remember that $coef_x = A/f_x^2$ )
- Using set 0, value $CAP_x$ can be interpolated.
- With $CAP_x$, value Y0 and Y1 can be interpolated from set 0 and set 1
- The FM_OSC value $OSC_x$ can be interpolated from Y0 and Y1.

The calculated value $OSC_x$ needs to be corrected because we approach the RF frequency with linear characteristic. The value of $OSC_x$ can be adjusted as follows:

- Write $CAP_x$ , $OSC_x$ to the LV2400x
- Measure the RF frequency $f_m$ at this point
- Calculate the $coef_m$ from $f_m$
- Determine the correction factor ($coef_x – coef_m$)
- Apply the correction factor to Y0 and Y1
- Interpolate the $OSC_x$ value again from the corrected Y0 and Y1

So to set RF frequency, we only need 1 measurement (if precision is required)
See APPENDIX E: CALCULATE CAP/OSC VALUE for pseudo code.

The 4 point P00, P01, P10, P11 can be measured once when the software is initialized
(See APPENDIX D: INITIALIZE THE QUICK SET FREQUENCY DATA)

# IMPLEMENTING RADIO FUNCTIONS

## Initialization

After power-up, the LV2400x needs to be initialized as follow:

1. Write default values to the registers:

| Block | Address | Register name | Value | Remark |
|---|---|---|---|---|
| 01h | 02h | MSRC_SEL | 40h | No measure source select, AFC at 20 dBμV |
| | 03h | FM_OSC | 80h | Indicative - should be tuned to FM frequency |
| | 04h | SD_OSC | 80h | Indicative - should be tuned to stereo decoder default frequency |
| | 05h | IF_OSC | 80h | Indicative – should be tuned to default IF frequency |
| | 06h | CNT_CTRL | 08h | User counter 1 without counter swapping |
| | 08h | IRQ_MSK | 00h | Disable all interrupts |
| | 09h | FM_CAP | 80h | Indicative – should be tuned to FM frequency |
| | 0Fh | IRQ_OUT | - | No action on this register (skip) |
| | | | | |
| 02h | 02h | RADIO_CTRL1 | 53h | Enable AFC. Reserved bits with correct value |
| | 03h | IF_CENTER | W | Same value as IF_OSC |
| | 05h | IF_BW | W | 65% of IF_OSC |
| | 06h | RADIO_CTRL2 | 10h | Mute IF-PLL inactive. VREF on |
| | 07h | RADIO_CTRL3 | 88h | Enable FM. Audio muted. AGC setlevel on |
| | 08h | STEREO_CTRL | 48h | Enable auto slew rate. CS level = 4 |
| | 09h | AUDIO_CTRL1 | 77h | Audio volume = 8. Tone level = 8 |
| | 0Ah | AUDIO_CTRL2 | C0h | No beep out |
| | 0Bh | PW_SCTRL | 6Dh | Power on. Soft streo=3. Soft mute = 3 |

2. Calibrate the IF frequency at **110 kHz** as follows:
   - Enable measure mode of LV2400X (register RADIO_CTRL1 – EN_MEAS bit)
   - Enable the demodulator PLL mute (register RADIO_CTRL2 – IF_PM_L bit)
   - Enable measuring IF-frequency (register MSRC_SELL – MSS_IF bit)
   - Tune the IF_OSC register to the specified frequency. Note that when writing to IF_OSC register, IF_CENTER register should be written with the same value of IF_OSC, IF_BW register should be written with IF_OSC*65%.
   - Restore measurement source select (register MSRC_SEL)
   - Disable the demodulator PLL mute (register RADIO_CTRL2 – IF_PM_L bit)
   - Restore the measure mode of LV2400X (register RADIO_CTRL1 – EN_MEAS bit)
3. Calibrate the stereo decoder clock at **38.3 kHz** as follows:
   - Enable measure mode of LV2400X (register RADIO_CTRL1 – EN_MEAS bit)
   - Enable the stereo PLL mute (register STEREO_CTRL - SD_PM bit)
   - Enable measuring stereo decoder frequency (register MSRC_SELL – MSS_SD bit)
   - Tune the SD_OSC register to the specified frequency.
   - Restore measurement source select (register MSRC_SEL)
   - Disable the stereo PLL mute (register STEREO_CTRL - SD_PM bit)
   - Restore the measure mode of LV2400X (register RADIO_CTRL1 – EN_MEAS bit)

**Note**: The audio should be un-muted after initialization.

***Software initialization***

- Initialize the global software data
  See APPENDIX D: INITIALIZE THE QUICK SET FREQUENCY
- Setting the radio region: initialize the FM band limit for scanning, set the de-emphasis
- Restore last user's settings like last radio station, stereo/mono… (this step is optional)

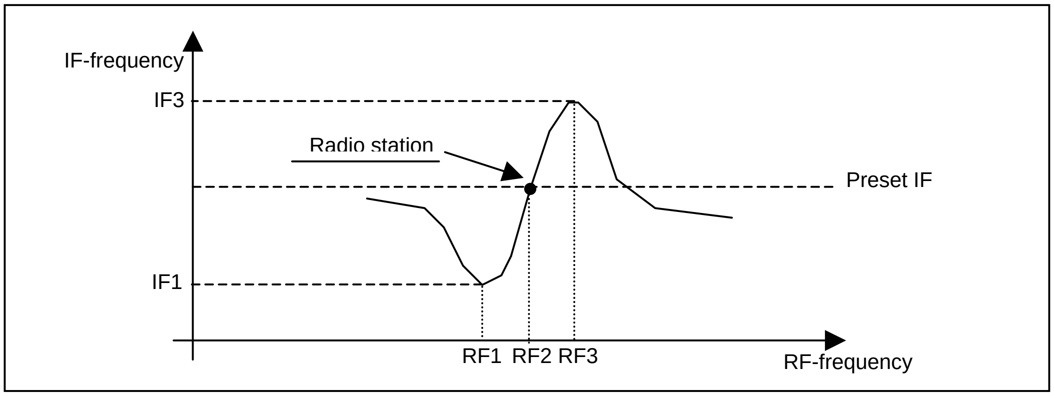
HINT: Audio can be muted during initialization.

## Setting frequency

- Calculate the RF frequency from the displayed frequency
- Disable AFC
- Mute the audio to hide noises during tuning process
- Enable measure mode of LV2400x (clear EN_MEAS# bit of Control Register A)
- Select OS_FM_OSC as output (OUTPUT_SELECT bits in control register A)
- Calculate the FM_CAP and FM_OSC (see APPENDIX E: CALCULATE CAP/OSC VALUE)
- Restore the output select of control register A
- Restore the measure mode of LV2400x
- Restore the audio mute state
- Restore AFC state

## Scan radio station

A radio station is characterized by the S-curve of IF-frequency (see picture below): when the RF-frequency is at F1, the IF-frequency is lower than the preset IF, and when the RF-frequency is at F3, the IF-frequency is higher than the preset IF. The difference RF-IF when RF varies from F1 to F3 remains the same: that is the displayed frequency of the radio station. The range F3-F1 depends on the strength of the radio station.



Scanning for a radio station is done in 2 phases:

1. Detecting the IF edge with sufficient delta IF (the swing between IF1 and IF3): Vary the RF-frequency until 2 points with sufficient IF-swing are measured.
2. Validating the IF edge with the knowledge that increasing RF will decrease IF (and vice versa), and the equation RF1 - IF1 = RF2 - IF2 (=displayed frequency) must be valid for a RF-frequency can be concluded as a radio station.

See APPENDIX G: SCAN RADIO STATION for complete scan description.

## Register map

The LV2400x registers are divided in 2 blocks:

| | |
|---|---|
| Block 01h | Status and measurement |
| Block 02h | Control |

To access a register in a block, the block must be first selected by writing the block number to the BLK_SEL register. Block selection can be skipped for subsequent accesses to other registers in the same block.

The mapping is as follows:

| Block | Address | Register name | Access | Operation |
|---|---|---|---|---|
| 01h | 00h | CHIP_ID | R | Chip identification |
| | 01h | BLK_SEL | W | Block Select |
| | 02h | MSRC_SEL | W | Measure source select |
| | 03h | FM_OSC | W | DAC control for FM-RF oscillator |
| | 04h | SD_OSC | W | DAC control for stereo decoder oscillator |
| | 05h | IF_OSC | W | DAC control for IF oscillator |
| | 06h | CNT_CTRL | W | Counter control |
| | 07h | NA | - | - |
| | 08h | IRQ_MSK | W | Interrupt mask |
| | 09h | FM_CAP | W | CAP bank control for RF-frequency |
| | 0Ah | CNT_L | R | Counter value low byte |
| | 0Bh | CNT_H | R | Counter value high byte |
| | 0Ch | CTRL_STAT | R | Control status |
| | 0Dh | RADIO_STAT | R | Radio station status |
| | 0Eh | IRQ_ID | R | Interrupt identify |
| | 0Fh | IRQ_OUT | W | Set Interrupt on DATA-line |
| | | | | |
| 02h | 01h | BLK_SEL | W | Access register 01h of block 1 |
| | 02h | RADIO_CTRL1 | W | Radio control 1 |
| | 03h | IF_CENTER | W | IF Center Frequency |
| | 04h | NA | - | - |
| | 05h | IF_BW | W | IF Bandwidth |
| | 06h | RADIO_CTRL2 | W | Radio Control 2 |
| | 07h | RADIO_CTRL3 | W | Radio control 3 |
| | 08h | STEREO_CTRL | W | Stereo Control |
| | 09h | AUDIO_CTRL1 | W | Audio Control 1 |
| | 0Ah | AUDIO_CTRL2 | W | Audio Control 1 |
| | 0Bh | PW_SCTRL | W | Power and soft control |

Not mentioned registers are not defined and should not be accessed.

## Register description

### *Block x, Register 01h – BLK_SEL – Block Select register (Write Only)*

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| BN[7:0] | | | | | | | |

Bit 7-0: **BN[7:0]:** 8-bit block number. For LV2400x, the following numbers are valid:
01h.
02h.

**Note:** This register can be accessed from any block

### *Block 1, Register 00h – CHIP_ID – Chip identify register (Read Only)*

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| ID[7:0] | | | | | | | |

Bit 7-0: **ID[7:0]:** 8-bit chip ID. The following ID's are defined:
05h for LV24000/LV24001
04h for LV24002

### *Block 1, Register 02h – MSRC_SEL – Measurement Source Select Register (Write-only)*

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| MSR_O | AFC_LVL | AFC_SPD | Reserved | Reserved | MSS_SD | MSS_FM | MSS_IF |

Bit 7: **MSR_O:** Output measure source to DATA-pin
0 = Measuring source not available at DATA-pin (normal operation).
1 = Measuring source available at DATA-pin (test mode).

Bit 6: **AFC_LVL:** AFC trigger level
0 = AFC is always active (trigger at 0 dBμV)
1 = AFC is only active when field strength is above 20 dBμV

Bit 5: **AFC_SPD:** AFC speed
0 = AFC adjusts with 3 Hz speed
1 = AFC adjusts with 8 kHz speed (test mode)

Bit 4: **Reserved:** Must be programmed with 0.

Bit 3: **Reserved:** Must be programmed with 0.

Bit 2: **MSS_SD:** Stereo decoder oscillator measurement
0 = Disable stereo decoder oscillator measurement
1 = Enable stereo decoder oscillator measurement

Bit 1: **MSS_FM:** FM RF oscillator measurement
0 = Disable FM RF oscillator measurement
1 = Enable FM RF oscillator measurement

Bit 0: **MSS_IF:** IF oscillator measurement
0 = Disable IF oscillator measurement
1 = Enable IF oscillator measurement

**Note:**

- Only one of the measurement source MSS_xx bits may be set at a time.
- The FM RF frequency is divided by 256 before it goes to the measuring circuitry.

### *Block 1, Register 03h – FM_OSC – FM RF Oscillator Register (Write-only)*

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| FMOSC[7:0] | | | | | | | |

Bit 7-0: **FMOSC[7:0]:** DAC value to control the FM RF oscillator (fine step)

**Note:**

- Positive DAC control (i.e. the frequency increases with the register’s value)
- See also FM_CAP register

### *Block 1, Register 04h – SD_OSC – Stereo Decoder Oscillator Register (Write-only)*

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SDOSC[7:0] | | | | | | | |

Bit 7-0: **SDOSC[7:0]:** DAC value to control the stereo decoder oscillator

**Note:** Positive DAC control (i.e. the frequency increases with the register's value)

### *Block 1, Register 05h – IF_OSC – IF Oscillator Register (Write-only)*

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| IFOSC[7:0] | | | | | | | |

Bit 7-0: **IFOSC[7:0]:** DAC value to control the IF oscillator

**Note:** Positive DAC control (i.e. the frequency increases with the register's value)

### *Block 1, Register 06h – CNT_CTRL – Counters Control Register (Write-only)*

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| CNT1_CLR | CTAB2 | CTAB1 | CTAB0 | SWP_CNT_L | CNT_EN | CNT_SEL | CNT_SET |

Bit 7: **CNT1_CLR:** Clear counter 1 bit
0 = Normal mode
1 = Clear and keep counter 1 in reset mode

Bit 6-4: **CTAB[2:0]:** Tab select for counter 2 measuring interval bits

| Value | Dec. | Stop value |
|---|---|---|
| 000b | 0 | Stop after 2 counts |
| 001b | 1 | Stop after 8 counts |
| 010b | 2 | Stop after 32 counts |
| 011b | 3 | Stop after 128 counts |
| 100b | 4 | Stop after 512 counts |
| 101b | 5 | Stop after 2048 counts |
| 110b | 6 | Stop after 8192 counts |
| 111b | 7 | Stop after 32768 counts |

Bit 3: **SWP_CNT_L:** Swap counter 1 and counter 2 bit (Active low)
0 = Clock source 1 to counter 2, clock source 2 to counter 1 (swapping)
1 = Clock source 1 to counter 1, clock source 2 to counter 2 (no swap)

Bit 2: **CNT_EN:** Enable the currently selected counter bit
0 = Disable counter (stop counting)
1 = Enable counter (counting mode)

Bit 1: **CNT_SEL:** counter select bit
0 = Select counter 1 for measurement
1 = Select counter 2 for measurement

Bit 0: **CNT_SET:** Set counters bit
0 = Normal mode
1 = Set both counter 1 and counter 2 to FFFFh and keep them set

***Block 1, Register 08h – IRQ_MSK – Interrupt Mask Register (Write-only)***

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | IM_MS | Reserved | Reserved | IRQ_LVL | IM_AFC | IM_FS | IM_CNT2 |

Bit 7: **Reserved:** Must be programmed with 0.

Bit 6: **IM_MS:** Mono/Stereo interrupt mask bit
0 = Disable mono/stereo change interrupt
1 = Enable mono/stereo change interrupt

Bit 5: **Reserved:** Must be programmed with 0.

Bit 4: **Reserved:** Must be programmed with 0.

Bit 3: **IRQ_LVL:** Interrupt level select bit
0 = Drive DATA-line from low to high when interrupt occurs (active high)
1 = Drive DATA-line from high to low when interrupt occurs (active low)

Bit 2: **IM_AFC:** AFC out of range interrupt mask bit
0 = Disable AFC out of range interrupt
1 = Enable AFC out of range interrupt

Bit 1: **IM_FS:** Field strength change interrupt mask bit
0 = Disable field strength change interrupt
1 = Enable field strength change interrupt

Bit 0: **IM_CNT2:** Counter 2 counting done interrupt mask bit
0 = Disable counter 2 counting done interrupt
1 = Enable counter 2 counting done interrupt

***Block 1, Register 09h – FM_CAP – FM RF Capacitor Bank Register (Write-only)***

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| FMCAP[7:0] | | | | | | | |

Bit 7-0: **FMCAP[7:0]:** CAP bank value to control the FM RF frequency (coarse steps)

**Note:**

- 7½ bit CAP value (Bit[7:6]: Combination 10b and 01b results in the same CAP-range)
- Negative control: de RF frequency decreases when increasing the register's value
- See also FM_OSC register

***Block 1, Register 0Ah – CNT_L – Counter Value Low Register (Read-only)***

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| CNT_LSB[7:0] | | | | | | | |

Bit 7-0: **CNT_LSB[7:0]:** Lower 8-bit value of the 16 bit counter

***Block 1, Register 0Bh – CNT_H – Counter Value High Register (Read-only)***

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| CNT_MSB[7:0] | | | | | | | |

Bit 7-0: **CNT_MSB[7:0]:** Upper 8-bit value of the 16 bit counter

***Block 1, Register 0Ch – CTRL_STAT – Control Status Register (Read-only)***

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | AFC_FLG |

Bit 7: **Reserved:** should be read as 0
Bit 6-1: **Reserved[6:1]:** should be read as all 1

Bit 0: **AFC_FLG:** AFC out of range bit
0 = AFC is within control range
1 = AFC is out of control range

**Note:** Reading this register will clear AFC, count 2 done interrupt.

***Block 1, Register 0Dh – RADIO_STAT – Radio Station Status Register (Read-only)***

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| RSS_MS | RSS_FS | | | | | | |

Bit 7: **RSS_MS:** Radio station mono/stereo state bit
0 = Mono
1 = Stereo

Bit 6-0: **RSS_FS[6:0]:** Radio station field strength bits
1111111b = Field strength less then 10 dBµV
0111111b = Field strength between 10 to 20 dBµV
0011111b = Field strength between 20 to 30 dBµV
0001111b = Field strength between 30 to 40 dBµV
0000111b = Field strength between 40 to 50 dBµV
0000011b = Field strength between 50 to 60 dBµV
0000001b = Field strength between 60 to 70 dBµV
0000000b = Field strength above 70 dBµV

**Note:** Reading this register will clear field strength and mono/stereo interrupt.

***Block 1, Register 0Eh – IRQ_ID – Interrupt Identify Register (Read-only)***

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | Reserved | II_CNT2 | Reserved | II_AFC | Reserved | Resweved | II_FS_MS |

Bit 7: **Reserved:** should be read as 1

Bit 6: **Reserved:** should be read as 1

Bit 5: **II_CNT2:** Counter 2 counting done flag
0 = No counting 2 counting done interrupt
1 = Measuring with counter 2 is done

Bit 4: **Reserved:** should be read as 0

Bit 3: **II_AFC:** AFC out of range interrupt bit
0 = No AFC interrupt
1 = AFC fails to hold the RF-frequency in range

Bit 2: **Reserved:** should be read as 0

Bit 1: **Reserved:** shold be read as 0

Bit 0: **II_FS_MS:** Field strength and Mono/Stereo interrupt bit
0 = No change in either the field strength or the mono/stereo mode
1 = Change in field strength bits detected or mono/stereo mode has changed

***Block 1, Register 0Fh – IRQ_OUT – Set Interrupt Out Register (Write Only)***

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| IRQO_VAL[7:0] | | | | | | | |

Bit 7-0: **IRQO_VAL[7:0]:** Write any value to this register will select the interrupt as output on the DATA-line of the LV2400x (the DATA-line can then be used as interrupt pin)

### *Block 2, Register 02h – RADIO_CTRL1 – Radio Control 1 Register (Write-only)*

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| EN_MEAS | EN_AFC | Reserved | Reserved | DIR_AFC | RST_AFC | Reserved | Reserved |

Bit 7: **EN_MEAS:** Enable measurement bit
0 = Normal mode
1 = Measurement mode

Bit 6: **EN_AFC:** Enable AFC bit
0 = Disable AFC
1 = Enable AFC
Note: Disable AFC in AM-radio mode.

Bit 5: **Reserved:** should be written with **0**

Bit 4: **Reserved:** should be written with **1**

Bit 3: **DIR_AFC:** AFC direction bit
0 = AFC normal direction
1 = AFC reverse direction (for test purpose)

Bit 2: **RST_AFC:** Reset AFC bit
0 = Normal operation
1 = Reset AFC to the middle of the control range

Bit 1: **Reserved:** should be written with **1**

Bit 0: **Reserved:** should be written with **1**

### *Block 2, Register 03h – IF_CENTER – IF Center Frequency Register (Write-only)*

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| IFCOSC[7:0] | | | | | | | |

Bit 7-0: **IFCENT[7:0]:** value for centering the IF frequency

### *Block 2, Register 05h – IF_BW – IF Bandwidth Register (Write-only)*

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| IFBW[7:0] | | | | | | | |

Bit 7-0: **IFBW[7:0]:** Value for IF bandwidth

### *Block 2, Register 06h – RADIO_CTRL2 – Radio Control 2 Register (Write-only)*

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| VREF2 | VREF | STABI_BP | IF_PM_L | Reserved | Reserved | AGCSP | AM_ANT_BSW |

Bit 7: **VREF2:** $V_{REF2}$ control bit
0 = $V_{REF2}$ is ON
1 = $V_{REF2}$ is OFF

Bit 6: **VREF:** $V_{REF}$ control bit
0 = $V_{REF}$ is ON
1 = $V_{REF}$ is OFF

Bit 5: **STABI_BP:** Stabi Bypass bit
0 = Internal voltage is Vstabe(normal operation)
1 = Internal voltage is Vcc (stabi bypassed)

Bit 4: **IF_PM_L:** IF PLL mute bit
0 = IF PLL mute on (presetting IF mode)
1 = IF PLL mute off (normal operation mode)

Bit 3: **Reserved:** should be written with 0

Bit 2: **Reserved:** should be written with 0

Bit 1: **AGCSP:** AGC speed control bit
0 = Normal speed
1 = High speed

**Note:** Turn on this bit will speed up the field strength measurement (fast tuning)

Bit 0: **Reserved:** should be written with 0

### *Block 2, Register 07h – RADIO_CTRL3 – Radio Control 3 Register (Write-only)*

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| AGC_SLVL | VOLSH | TB_ON | AMUTE_L | SE_FM | Reserved | SE_BE | SE_EXT |

Bit 7: **AGC_SLVL:** AGC setlevel bit
This bit must be set to 1 for normal operation mode.

Bit 6: **VOLSH:** Volume level shift bit
0 = Normal volume level
1 = Extra volume of 12 dB

Bit 5: **TB_ON:** Treble/Bass on bit
0 = Turn off treble/Bass control
1 = Turn on treble/Bass control

**Note:** This bit should be written with 1 when one of the TREB_N, TREB_P, BASS_N, BASS_P and BASS_PP bits of AUDIO_CTRL2 register is 1. When none of these bits is set, this bit should be written with 0

Bit 4: **AMUTE_L:** Audio mute bit
0 = Audio muted
1 = Audio not muted

Bit 3: **SE_FM:** FM radio select bit
0 = Disable FM radio
1 = Enable FM radio

Bit 2: **Reserved:** should be written with 0

Bit 1: **SE_BE:** Beep select bit
0 = Disable beep
1 = Enable beep

**For LV24000:**
Bit 0: **Reserved:** should be written with 0

**For LV24001 and LV24002:**
Bit 0: **SE_EXT:** External source select bit
0 = Disable external source
1 = Enable external source

### *Block 2, Register 08h – STEREO_CTRL – Stereo Control Register (Write-only)*

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| FRCST | FMCS[2:0] | | | AUTOSSR | DISITG | SD_PM | ST_M |

Bit 7: **FRCST:** Force stereo bit
0 = Normal mode
1 = Force stereo mode for test

Bit 6-4: **FMCS[2:0]:** FM channel separation bits
0…7 = FM channel separation level

Bit 3: **AUTOSSR:** Auto stereo slew rate enable bit
0 = Disable stereo auto slew rate
1 = Enable stereo auto slew rate

Bit 2: **DISITG:** Disable integrator bit
0 = Enable integrator
1 = Disable integrator

Bit 1: **SD_PM:** Stereo decoder PLL mute bit
0 = Stereo decoder PLL not muted (normal operation)
1 = Stereo decoder PLL is muted (presetting mode)

Bit 0: **ST_M:** FM stereo/mono mode bit
0 =Stereo mode
1 = Mono mode

### *Block 2, Register 09h – AUDIO_CTRL1 – Audio Control 1 Register (Write-only)*

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| TONE_LVL | | | | VOL_LVL | | | |

Bit 7-4: **TONE_LVL:** Tone level bits
1111b = Minimum tone level.
0000b = Maximum tone level.

Bit 3-0: **VOL_LVL:** volume level bits
1111b = Minimum volume level.
0000b = Maximum volume level.
Each level is 3dB volume adjustment.

**Note:** The tone level may not be greater than the volume level. This means the value of bit [7:4] must be greater or equal to the value of bit [3:0] (the higher the value, the lower the level)

### *Block 2, Register 0Ah – AUDIO_CTRL2 – Audio Control 2 Register (Write-only)*

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| BPFREQ | | DEEMP | TREB_N | TREB_P | BASS_N | BASS_P | BASS_PP |

Bit 7-6: **BPFREQ:** Beep frequency bits
00b = 2 kHz beep tone.
01b = 1 kHz beep tone.
10b = 0.5 kHz beep tone.
11b = beep-output high.

**Note:** Bit [7:6] should be programmed with 11b when beep source is disabled (SE_BE bit of RADIO_CTRL3 register is 0)

Bit 5: **DEEMP:** De-emphasis bit
0 = De-emphasis 50 µs.
1 = De-emphasis 75 µs.

Bit 4: **TREB_N:** Treble negative bit
0 = Normal treble
1 = Negative treble

Bit 3: **TREB_P:** Treble positive bit
0 = Normal treble
1 = Positive treble

**Note:** TREB_N and TREB_P may be not be activated at the same time.

Bit 2: **BASS_N:** Bass negative bit
0 = Normal bass
1 = Negative bass

Bit 1: **BASS_P:** Bass positive bit
0 = Normal bass
1 = Positive bass

**Note:** BASS_N and BASS_P may be not be activated at the same time.

Bit 0: **BASS_PP:** Bass extra positive level bit
0 = Normal bass positive level
1 = Extra bass positive level

***Block 2, Register 0Bh – PW_SCTRL – Power and Soft Control Register (Write-only)***

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SS_CTRL | | | SM_CTRL | | | PW_HPA | PW_RAD |

Bit 7-5: **SS_CTRL:** Soft stereo control bits (8 levels)
000b = Minimal soft stereo (off)
111b = Maximal soft stereo level

Bit 4-2: **SM_CTRL:** Soft audio mute bits (8 levels)
000b = Minimal audio mute (off)
111b = Maximal soft audio mute level

**For LV24000 and LV24001:**

Bit 1: **Reserved:** should be written with 0

**For LV24002:**

Bit 1: **PW_HPA:** Headphone amplifier power bit
0 = Headphone amplifier is switched OFF.
1 = Switch headphone amplifier ON

**Note:** PW_HPA is 0 at power up

Bit 0: **PW_RAD:** Radio circuitry power bit
0 = Radio circuitry is switched OFF.
1 = Switch radio circuitry ON

**Note:**

- PW_RAD is 0 at power up
- PW_RAD does not switch on the headphone amplifier of the LV24002. The headphone amplifier is controlled by PW_HPA bit.

# APPENDIX A: NOTATIONS USED BY PSEUDO-CODE

| | |
|---|---|
| *//* | *Comment* |
| *=* | *Assignment* |
| *==* | *Test for equal* |
| *!=* | *Test for not equal* |
| *>* | *Test for greater* |
| *<* | *Test for smaller* |
| *>>* | *Shift right (bit wise)* |
| *<<* | *Shift left (bit wise)* |
| *abs* | *absolute function (example: abs(1-2)=1)* |

# APPENDIX B: BIG STEP TUNING

```
Input:
        x1: Low value to write to the DAC control register (first point  of the interpolating)
        x2: high value to write the DAC control register (Second point of the interpolating)
        f: frequency to be set

Write x1 to the target DAC control register
f1 = Measure frequency at x1
Loop
        Write x2 to the target DAC control register
        f2 = Measure frequency at x2
        Check f2 against f – terminate loop if f2 within margin
        step = InterpolatingX(f, x1, x2, f1, f2)
        if step is 0 terminate loop // can not approach the frequency with interpolating
        x_new = x1 + step             // Caution: step can be negative!
        If point2 closer to expected frequency,Move point2 to point1 by assigning x1=x2; f1=f2
        Make new point2 by assigning x2 = x_new     // f2 will be measured
        Repeat loop
```

*NOTE:*

- *The interpolating routine interpolatingX is described in* APPENDIX H: MISCELLANEOUS ROUTINES
- *For LV2400x: the logical value 10 can be used for x1, 240 for x2*
- *Write to DAC control register: the algorithm works with logical values. Apply conversion to physical value if necessary.*

# APPENDIX C: FINE STEP TUNING

```
Input:
        Initial step: any value > 0
        Register Value: Start value of a DAC control register for the fine tuning
        f: frequency to be set

step = initial step
Loop
        Register Value = Register Value + step            // step can be negative!
        Write Register Value to the target DAC control register
        f1 = Measure Frequency at Register Value
        Check f1 against f – terminate loop if f1 within margin
        If f1<f
                step = absolute(step)         // increase register value
                set TOO_SMALL flag
                if step == 1 also set APPROACH_UP_1 flag
        If f1>f
                step = -absolute(step)        // decrease register value
                set TOO_BIG flag
                if step= = -1 also set APPROACH_DOWN_1 flag
        If both APPROACH_UP_1 and APPROACH_DOWN_1 flags set
                terminate loop      // approached with step=1: best fit value found
        else if both TOO_BIG and TOO_SMALL flags set
                step = step/2
                If step==0 make step=1
        repeat loop
```

*NOTE:*

- *Apply this routine when precision is required (getting the best fit value for a DAC control register)*
- *Initial Step can be set as 16 for LV2400x*
- *Write to DAC control register: the algorithm works with logical values. Apply conversion to physical value if necessary.*

# APPENDIX D: INITIALIZE THE QUICK SET FREQUENCY DATA

```
SwOscLow = 30
SwOscHigh = 200

SwCapLow = 0
SwCapHigh = 191

Write SwOscLow to the LV2400x

Write SwCapLow to the LV2400x
Measure f00
Coef00 = CalculateCoeff(f00)

Write SwCapHigh to the LV2400x
Measure f01
Coeff01 = CalculateCoeff(f01)
Write SwOscHigh to the LV2400x

Write SwCapLow to the LV2400x
Measure f10
Coef10 = CalculateCoeff(f10)

Write SwCapHigh to the LV2400x
Measure f11
Coeff11 = CalculateCoeff(f11)
```

*NOTE:*

- *Values Coef00, Coef01, Coef10, Coef11 should be stored for later usage*
- *The software CAP/OSC values are logical values. Apply conversion to physical values before writing them to the LV2400x.*

# APPENDIX E: CALCULATE CAP/OSC VALUE

*Input:* *the RF-frequency f*
*PrecisionLevel: NONE, LOW, MEDIUM, HIGH*
*Output: the calculated CAP, OSC value*

```
// Determine correction action
if (PrecisionLevel is NONE)
    MeasureTime = 0ms
    CorFreq = 0Hz

if (PrecisionLevel is LOW) OR (PrecisionLevel is MEDIUM)
    MeasureTime = 32ms
    CorFreq = 32Hz

if (PrecisionLevel is HIGH)
    MeasureTime = 64ms
    CorFreq = 16Hz
CorFreq = CorFreq * DividerFactor  // For RF –frequency: DividerFactor=256

Coef = CalculateCoeff(f)
CapValue = InterpolateX(Coef, SwCapLow, swCapHigh, Coef00, Coef01)
Write SwOscLow to LV2400x
Done = FALSE
While (Done is FALSE)
    CoefLo = InterpolateY(CapValue, SwCapLow, swCapHigh, Coef00, Coef01)
    CoefHi = InterpolateY(CapValue, SwCapLow, swCapHigh, Coef10, Coef11)
    CoefRange = CoefLo – CoefHi
    Write CapValue to the LV2400x

    If MeasureTime is not 0
        Fcur = Measure RF-frequency with MeasureTime
        CoefFcur = CalculateCoeff(Fcur)
        CoefCor = CalculateCoeff(Fcur + CorFreq)
        CoefLo = CoefFcur + CoefCor
        CoefHi = CoefCur – CoefRange - CoefCor
    else
        CoefCur = CoefLo

    if Coef is in range [CoefLo, CoefHi]
        OscValue = InterpolateX(Coef, SwOscLow, SwOscHigh, CoefLo, CoefHi)
        If OscValue is in range [swOscLow, SwOscHi] Done = TRUE

    if (Done is FALSE)
        CapNew = InterpolateX(Coef, CapValue, SwCapHigh, CoefCur, Coef01)
        If CapNew is equal to CapValue
            if Coef is smaller than CoefCur
                CapValue = CapValue + 1
            Else
                CapValue = CapValue – 1
        Else
            CapValue = CapNew
        Repeat the while-loop
```

*<u>NOTE</u>:*

- *The software CAP/OSC values are logical value. Apply conversion to physical values before writing them to the LV2400x.*
- *Interpolating is described in* APPENDIX H: MISCELLANEOUS ROUTINES

# APPENDIX F: QUICK SET RF-FREQUENCY

```
Input:    the RF-frequency f
          PrecisionLevel: NONE, LOW, MEDIUM, HIGH

Calculate CAP/OSC value for frequency f (see APPENDIX E: CALCULATE CAP/OSC VALUE)
If (PrecisionLevel is NONE) OR (PrecisionLevel is LOW)
          Write OscValue to LV2400x
          Exit

if (PrecisionLevel is MEDIUM)
          ValidateCapOsc with 8ms measurement
          Exit

if (PrecisionLevel is HIGH)
          ValidateCapOsc with 64ms measurement
          Exit
```

### ValidateCapOsc

```
Input:    the RF-frequency f
          MeasureTime

Retry = 0
Loop
          Write CapValue to LV2400x
          BigStepTuning (see APPENDIX B: BIG STEP TUNING)
          If FAILED
                    if f < Current RF
                              CapValue = CapValue + 1
                    Else
                              CapValue = CapValue – 1
                    Retry = Retry + 1
                    If Retry is greater than 3
                              Exit with unreachable frequency failure
                    Else
                              Repeat loop
          if OK
                    Done, exit loop
```

# APPENDIX G: SCAN RADIO STATION

*Input: The field strength level of the desired station UserFs*
*The scan direction: iDir*
*+1: scan up (increase RF-frequency)*
*-1: scan down (decrease RF-frequency)*

*Adjust the UserFs/Calculate ScanStep, IfSwing*

| UserFs | QuickFsLevel | ScanStep | IfSwing |
|---|---|---|---|
| 0 | 0 | 60 kHz | 40 kHz |
| 1 | 0 | 65 kHz | 45 kHz |
| 2 | 0 | 70 kHz | 50 kHz |
| 3 | 1 | 75 kHz | 55 kHz |
| 4 | 2 | 75 kHz | 55 kHz |
| 5 | 3 | 75 kHz | 55 kHz |
| 6 | 4 | 75 kHz | 55 kHz |
| 7 | 5 | 75 kHz | 55 kHz |

### *Scan algorithm:*

```
If Current RF is a valid station
        Step = 200 kHz              // Step away from current station
Else
        Step = 0
IfSwing = abs(IfSwing) * iDir       // Apply scan direction as sign in IfSwing
IfSwing = IfSwing * -1              // Negate IfSwing because of IF phase
IF1 = 0
Rf = Current RF-frequency
ValidStation = FALSE

While ValidSation is FALSE
        Rf = Rf + (Step * iDir)
        If Check RF limit failed: exit with limit failure
        QuickSetFrequency(Rf, PrecisionLevel)
        If FAILED exit with unreachable frequency failure
        Update display if necessary
        if MeasuredFS is smaller than QuickFsLevel
        Step = ScanStep
        Repeat the While-loop
        IF2 = Measure IF-frequency with 8ms
        EdgeFound = FALSE

        If IF1 is not 0
                DeltaIF = IF1 – IF2
                If (IfSwing is greater than 0) AND (DeltaIF is greater than IfSwing)
                        EdgeFound = TRUE
                If (IfSwing is smaller than 0) AND (DeltaIF is smaller than IfSwing)
                        EdgeFound = TRUE

        IF1 = IF2           // Take over the IF for next time
        If EdgeFound is FALSE
                Step = ScanStep
                Repeat the while-loop

        // Correct edge: Pre-check the field strength
        if MeasuredFS is smaller than QuickFsLevel
                Step = ScanStep
                Repeat the while-loop

        // Pinpoint the radio station
        ValidStation = FindFmStation
```

```
    If ValidStation is FALSE
        Step = 100kHz
        Repeat the while-loop

    // Post-check field strength
    if MeasuredFS is smaller than UserFS
        Step = ScanStep
        ValidStation = FALSE
        Repeat the while-loop

    // Hereafter: exit while-loop with ValidStation is TRUE: station is found at current RF
```

### FindFmStation

```
IfFreq = Measure IF-frequency with 32ms
If IF within range [Preset IF ± 15kHz] exit with StationFound is TRUE
RfFreq = Measure RF-frequency with 32ms
Retry = 0
Loop
        RfFreq = RfFreq + (Preset IF – IfFreq)  // Caution: Preset IF - RfStep is signed and can be negative
    QuickSetFrequency to set RfFreq with precision is HIGH
    If FAILED exit with StationFound is FALSE
    IfFreq = Measure IF-frequency with 32ms
    If IF within range Preset IF ± 15kHz
        exit with StationFound is TRUE
    else
        Retry = Retry + 1
    If Retry reaches value 4
        Exit with StationFound is FALSE
    Repeat loop
```

*NOTE:*

- *Quick set frequency is described in* APPENDIX F: QUICK SET RF-FREQUENCY

# APPENDIX H: MISCELLANEOUS ROUTINES

## Constant A for RF-frequency coefficient

*When working with floating point is possible, the coefficient can be calculated as it is i.e. $1/f^2$.*
*But when it is difficult to work with floating point, constant A must be so chosen that it shifts the coefficients to integer values that are big enough to cover the frequency range 66 MHz-116 MHz without overflow. The resolution of the coefficient must be about 4 kHz to work with the RF-frequencies*
*Consider the following suggestion:*
*Derive from formula coef = $A/f^2$*

$$Coef = \frac{A}{f^2} = \frac{\frac{A1}{f} * A2}{f} * A3$$

*When A1, A2 and A3 are chosen as follows*

$A1 = 2^{32} - 1$
$A2 = 2^{16}$
$A3 = 2^{8}$

*The coefficients will fit in 32 bits integer.*

## CalculateCoeff

```
Input:    The RF-frequency f in Hz
Output:
          $f_{kHz} = f/1000$
          if ($f_{kHz}$ == 0)
                    Coef = 0
          else
                    Coef = ( (A1 / f) * A2) / f ) * A3
```

## InterpolateX

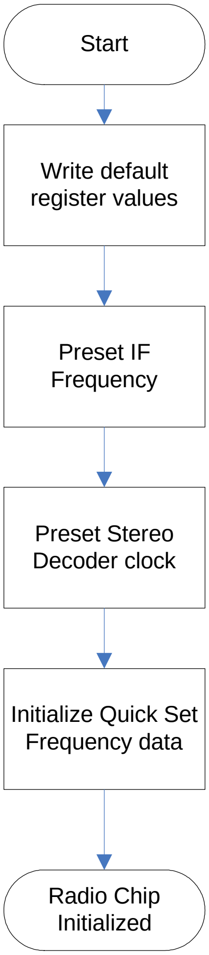
```
Input: ExpectedY, x1, x2, y1, y2
Output:
If y1 == y2
          X = 0
else
          X = (ExpectedY - y1)*(x2 - x1) / (y2 - y1) + x1
```

## InterpolateY

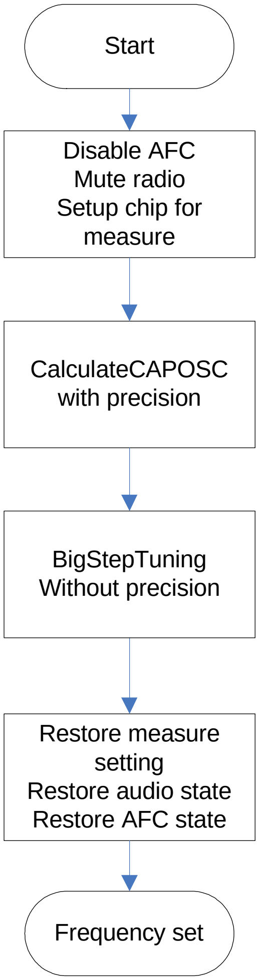
```
Input: ExpectedX, x1, x2, y1, y2
Output:
If x1 == x2
          Y = 0
else
          Y = (ExpectedX - x1)*(y2 - y1) / (x2 - x1) + y1
```

# APPENDIX G: FLOW CHARTS

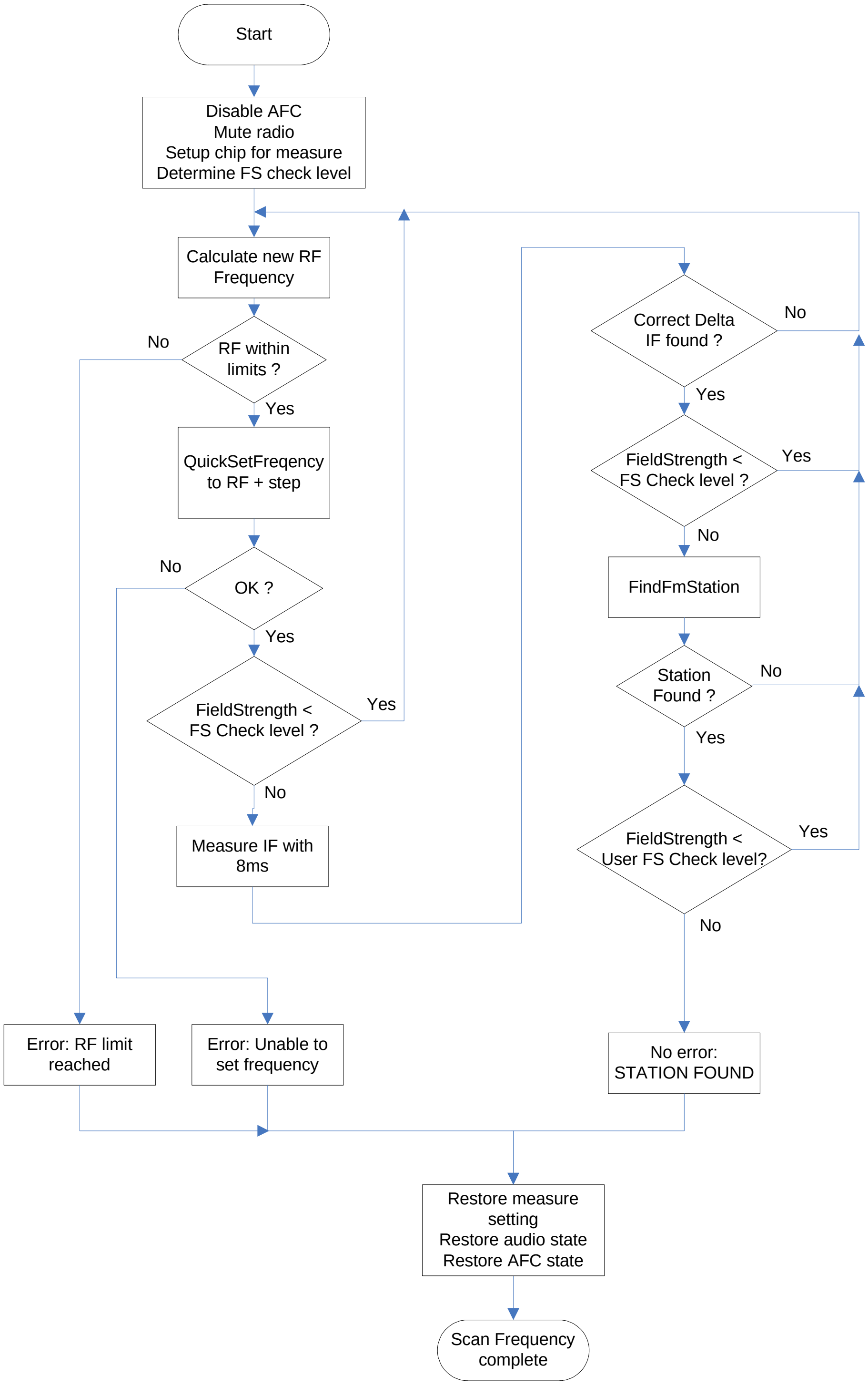
**Initialize Radio**


Start
Write default register values
Preset IF Frequency
Preset Stereo Decoder clock
Initialize Quick Set Frequency data
Radio Chip Initialized

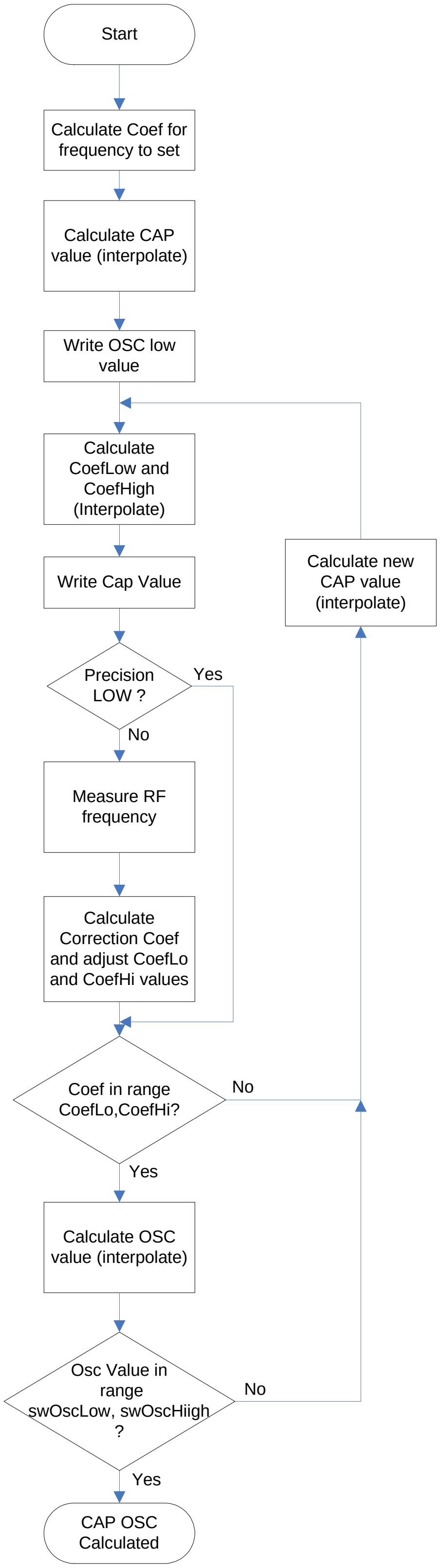
**Set Frequency**



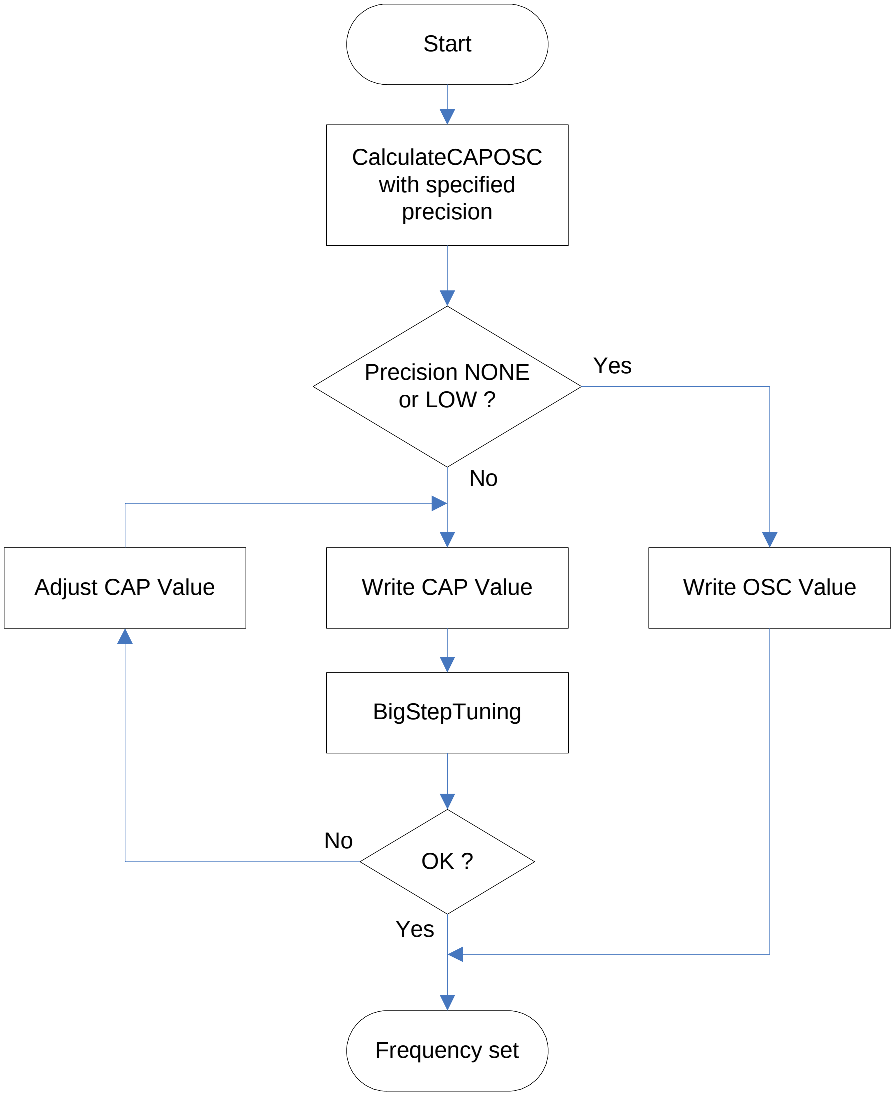
**Scan Frequency**



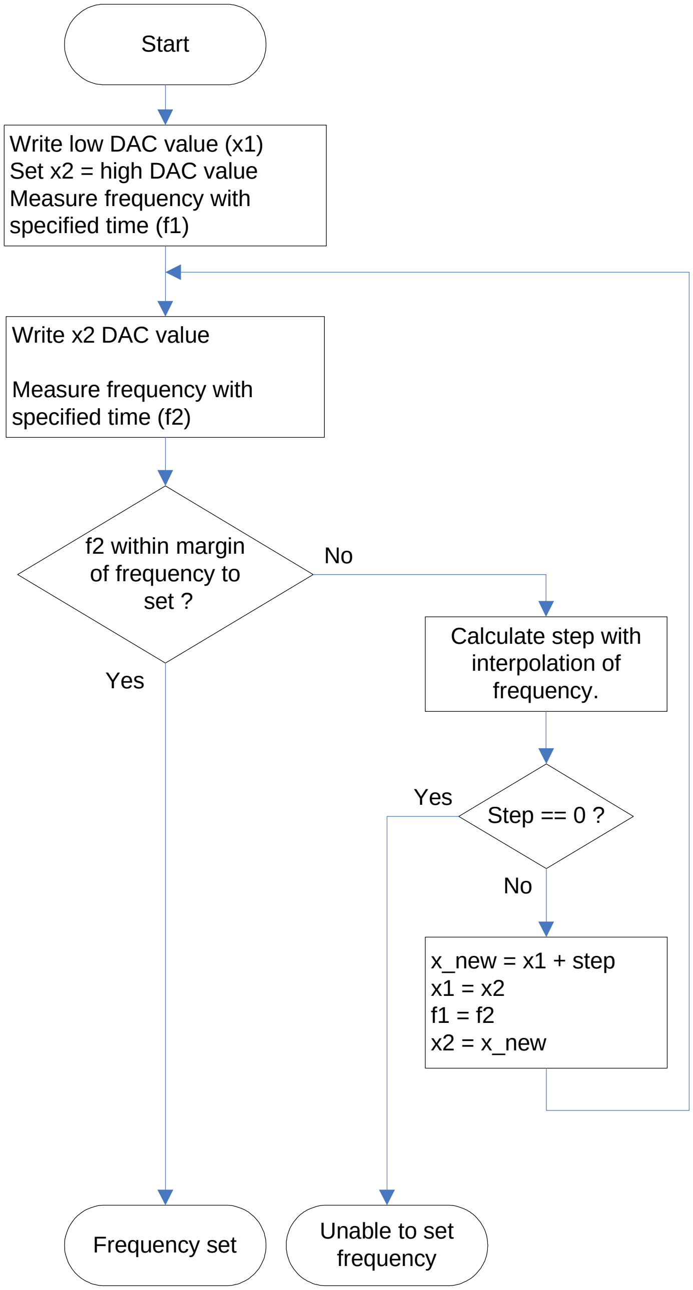
**Calculate CAP OSC**


Start
Calculate Coef for frequency to set
Calculate CAP value (interpolate)
Write OSC low value
Calculate CoefLow and CoefHigh (Interpolate)
Write Cap Value
Calculate new CAP value (interpolate)
Precision LOW ?
Yes
No
Measure RF frequency
Calculate Correction Coef and adjust CoefLo and CoefHi values
Coef in range CoefLo,CoefHi?
No
Yes
Calculate OSC value (interpolate)
Osc Value in range swOscLow, swOscHiigh ?
No
Yes
CAP OSC Calculated

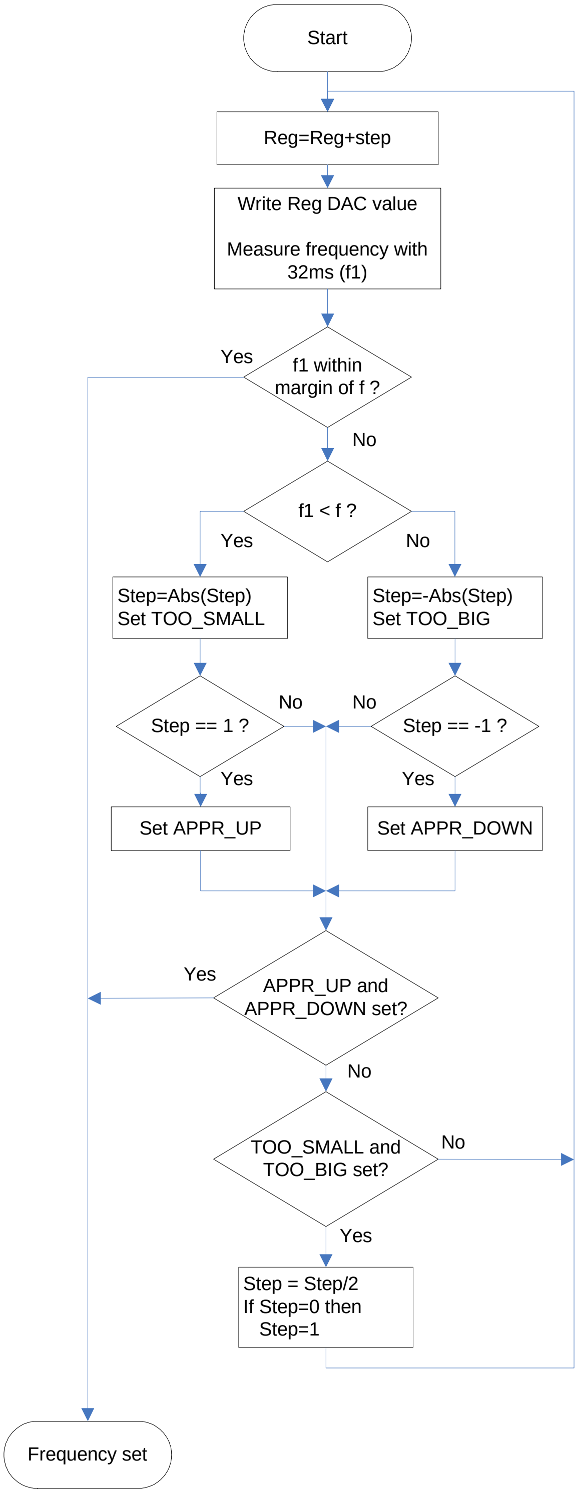
### QuickSetFrequency



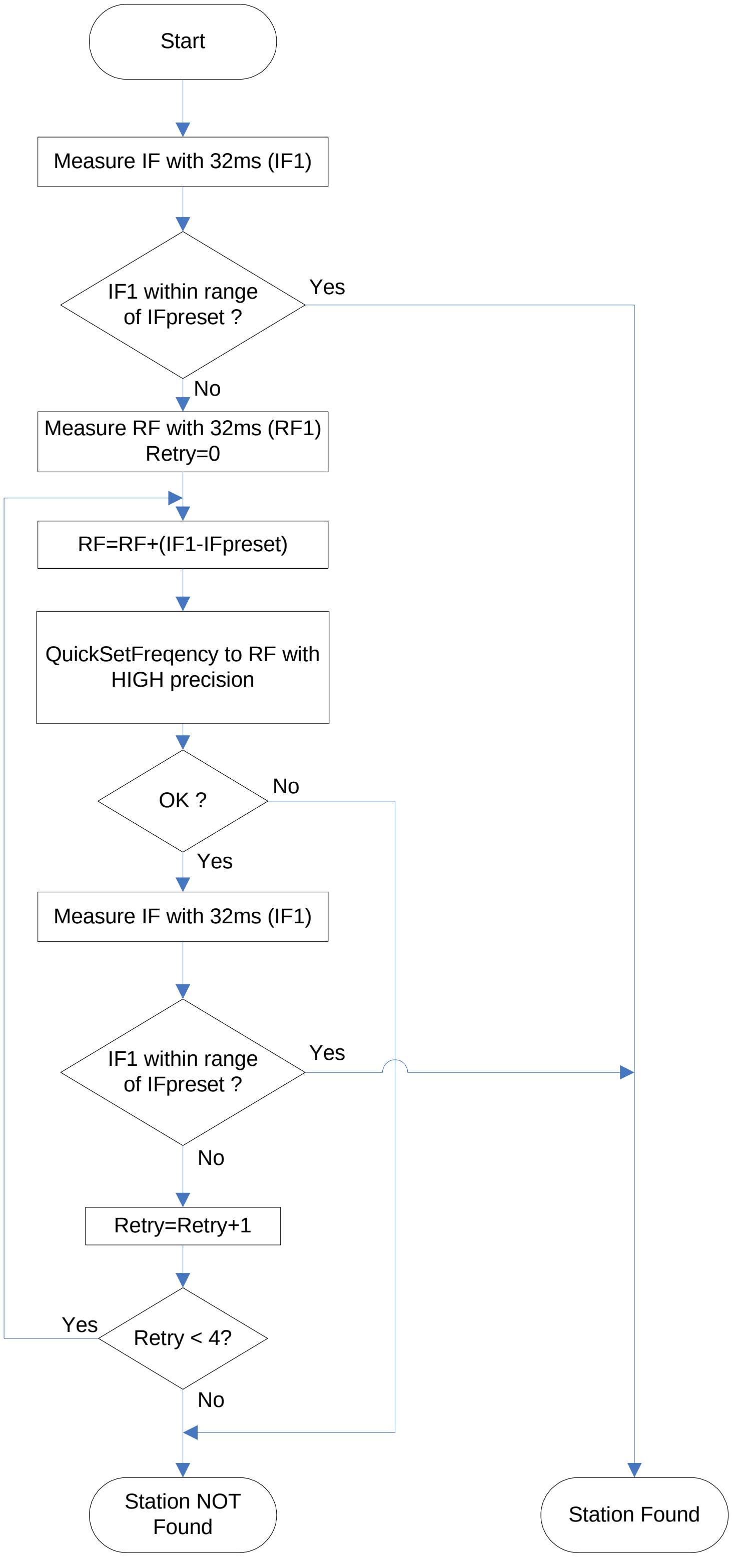
### BigStepTuning



### FineStepTuning



**FindFmStation**


Start
Measure IF with 32ms (IF1)
IF1 within range of IFpreset ?
Yes
No
Measure RF with 32ms (RF1)
Retry=0
RF=RF+(IF1-IFpreset)
QuickSetFreqency to RF with HIGH precision
OK ?
No
Yes
Measure IF with 32ms (IF1)
IF1 within range of IFpreset ?
Yes
No
Retry=Retry+1
Yes
Retry < 4?
No
Station NOT Found
Station Found